

TIS

TOTAL INFORMATION SERVICES

Workbook 2



PET String and Array Handling

CONTENTS

I.	INTRODUCTION.	1-1
	A. Assumptions Made About the User	1-1
	B. Notation Used in This Workbook.	1-1
II.	SUBSCRIPTED VARIABLES	2-1
	A. Legal Subscripts.	2-2
	B. Dimensions.	2-2
III.	STRINGS	3-1
	A. Legal String Names.	3-2
	B. Subscripted String Variables.	3-3
	C. Maximum String Length	3-3
IV.	OPERATIONS ON STRINGS	4-1
	A. Statements.	4-1
	1. INPUT	4-1
	2. DATA, READ, RESTORE	4-2
	3. GET	4-3
	B. Functions	4-3
	1. ASC	4-3
	2. CHR\$.	4-4
	3. LEFT\$.	4-5
	4. RIGHT\$.	4-6
	5. MID\$.	4-6
	6. STR\$.	4-8
	7. VAL	4-9
	8. LEN	4-10
V.	EXERCISES	5-1
VI.	APPENDIX - Character Sets	6-1

I. INTRODUCTION

This workbook gives a series of exercises that show the new user how to use the Commodore PET 2001 computer. The most effective way to use the workbook is to sit down with a PET and go through the exercises as they are presented. Enough space has been provided in the workbook for you to add your own examples as you develop them. Later, when you need to refresh your memory on a particular topic, these examples should supply pertinent, meaningful information.

A. Assumptions Made About the User

Some PET users are comfortable with mathematics. However this workbook assumes that the majority are not. For that reason, most exercises will use nothing more than high school arithmetic.

This workbook also assumes you know something about the syntax and semantics of the BASIC programming language. If you do not, we recommend that you obtain one of the following books:

BASIC Programming, J. Kemeny and T. Kurtz

BASIC, Albrecht, Finke, Brown

What Do You Do After You Hit Return? Peoples Computer Co.

Basic BASIC, James Coan

Advanced BASIC, James Coan

Read about BASIC syntax; then alternate between this workbook and your text on BASIC. That way you will learn both BASIC and how to use the PET.

B. Notation Used in This Workbook

We use a consistent notation in this workbook to indicate what is to be typed on the keyboard (T:), what appears on the TV display (R:), and what indicates blanks are to be typed (b). For example:

T: info ('RETURN') key

means to type the characters contained on the line after the colon (:) followed by a 'RETURN'.

R: response

means that the system response to the previous T: line should be a line on the TV.

INTRODUCTION

Blanks are important. They are specified by b. For example:

T: ?"ABbC" ('RETURN' key)

means type ?, then ", then the letter A, then the letter B, then a space, then the letter C, then " followed by a 'RETURN'. Now let's run that example all together:

T: ?"ABbC" ('RETURN' key)

R: AB C

The special keys on the PET keyboard can cause some confusion. This workbook identifies the special keys with the notation 'KEYNAME'. 'KEYNAME' means press the named key. The unquoted sequence of characters OTHER means, press the five keys O T H E R in succession.

Example: Special key notation

T: 'STOP'

means press the key labeled STOP.

T: STOP

R: BREAK

R: READY.

means press the 4 keys S T O P in succession.

II. SUBSCRIPTED VARIABLES

Just like a regular variable, a subscripted variable names a logical location in the computer's memory. These locations may be thought of as small, contiguous boxes, such as post office boxes, where the subscripted variable is the name of a box and the contents of the box is the value of the subscripted variable. Another way of thinking of these memory locations is as rows or columns. In the example below the memory locations are the boxes, the contents of the memory are the numbers in the boxes, and the address or name of the memory location is the subscripted variables on the left.

A(0)	1	The value of A(0) is 1
A(1)	3	The value of A(3) is 211
A(2)	10	
A(3)	211	The name of the location of 3 is A(1)

An array of one column or one row is called a one-dimensional array or vector. Two-dimensional arrays consist of both rows and columns, sometimes called a matrix. The subscripts define the row and the column.

	Col 1	Col 2	Col 3
Row 1	A(1,1)	A(1,2)	A(1,3)
Row 2	A(2,1)	A(2,2)	A(2,3)

Subscripted variables make keeping track of elements in an array or table much easier. In the PET BASIC, numeric, integer, and string variables may be subscripted. Using subscripted variables makes it easy to assign a lot of values to a lot of variables. The reason this is so easy is that subscripted variables work so well in FOR-NEXT loops. We will be using several examples where the subscripted variable is set to the value of the subscript.

Exercise: Input and output of non-subscripted and subscripted arrays.

```
T: NEW
T: 10 DATA 5,7,8,11,12
T: 20 READ A1,A2,A3,A4,A5
T: 30 PRINT A1;A2;A3;A4;A5
T: RUN
R: 5 7 8 11 12
```

SUBSCRIPTED VARIABLES

Without subscripted arrays you must write each variable separately. This isn't bad with only five variables but think of 100. The following code uses subscripts to do the same thing.

```
T: 20 FOR I = 1 TO 5
T: 30 READ A(I)
T: 40 PRINT A(I);
T: 50 NEXT I
T: RUN
R: 5 7 8 11 12
```

A. Legal Subscripts

Legal values for subscripts are non-negative integers. Variables, formulas, functions, and constants may be used for subscripts.

Exercise: Show that variables, functions, and formulas are legal subscripts.

```
T: NEW
T: 10 A=2
T: 20 B=3
T: 30 C=5
T: 40 FOR I=1 TO 10
T: 50 D(I)=I
T: 60 NEXT I
T: 70 PRINT D(A*B-C)
T: 80 PRINT D(INT(C/A))
T: 90 PRINT D(C-INT(B/A)+A)
T: RUN
R: 1
R: 2
R: 6
R: READY.
```

B. Dimensions

A subscripted variable does not need a corresponding DIM statement as long as the value of the subscript does not exceed 10 (this is true even in multiple subscripts). If you intend to use a subscript of 0, the maximum value of a subscript is still 10. So if you use 0 as a subscript you may have up to 11 for each subscripted variable without needing a DIM statement for that variable.

PET STRING AND ARRAY HANDLING

Exercise: Find the limit of non-dimensioned subscripted numeric variables.

```
T: 10 FOR J=1 TO 20
T: 20 FOR I=1 TO 20
T: 30 A%(I,J)=I
T: 40 PRINT A%(I,J);I;J
T: 50 NEXT I
T: 60 NEXT J
T: 70 END
T: RUN
R: 1 1 1
R: 2 2 1
.
.
.
R: 10 10 1
R: ? BAD SUBSCRIPT ERROR IN 30
```

Without a DIM statement for A% the system will not continue when either I or J exceeds 10.

Exercise: Demonstrate a non-dimensioned subscripted variable starting with 0.

```
T: NEW
T: 10 FOR I=0 TO 10
T: 20 A(I)=I
T: 30 PRINT A(I);I
T: 40 NEXT I
T: RUN
R: 0 0
R: 1 1
R: 2 2
.
.
.
R: 10 10
R: READY .
```

The following list is the maximum allowable DIMensions for an 8K PET without getting a ?OUT OF MEMORY ERROR. These were checked out using the calculator mode, which means there was no program in the machine. The maximum number of subscripts is 3.

DIM A(14,14)	DIM A(37,36)	DIM A(10,10,10)
DIM A\$(14,31)	DIM A\$(38,37)	DIM A\$(11,11,10)
DIM A%(14,32)	DIM A%(41,40)	DIM A%(14,14,14)

SUBSCRIPTED VARIABLES

If you observe that the maximum value of the 3-dimensional array for A is 10, it follows from previous examples that A would never need a DIM statement.

The previous list of dimensioned arrays is somewhat misleading. If the dimension of a variable exceeds 255, the array wraps around and starts over.

Exercise: Demonstrate the 255 limit on arrays.

T: DIM A(300)

T: A(0)=0:A(255)=255:PRINT A(0);A(255)

R: 0 255

T: A(256)=256:PRINT A(0);A(255);A(256)

R: 256 255 256

As you can see, the number 256 was stored in A(0).

Let's use an array and a subscripted variable ^{dobblester} to check the PET random-number generator by rolling a die. We'll let D(1) be the storage location when a 1 is rolled, D(2) the location for a 2, and so on. Let's roll the die 6000 times and see if each number has an equal probability of coming up.

Exercise: Check if the random number generator gives reasonable results throwing a die.

T: NEW

T: 10 FOR J=1 TO 6

T: 20 D(J)=0

T: 30 NEXT J

T: 40 FOR I=1 TO 6000

T: 50 K=INT(6*RND(1))+1

Generate random numbers 1 through 6. RND(1) generates random numbers between 0 and 1. A given random number may be equal to 0 but will always be less than 1.

T: 60 D(K)=D(K)+1

T: 70 NEXT I

T: 80 PRINT D(1);D(2);D(3);D(4);D(5);D(6)

T: RUN

R: 1015 1005 964 964 1050 1002

R: READY.

The run time for this many random numbers is over a minute. These numbers should vary each time the program is run. What does this prove?

III. STRINGS

Each computer system has its own character set, that is, a collection of symbols which the computer recognizes. These symbols may be digits, alphabetic letters, or special characters. In our case, we are talking about the symbols on the PET keyboard. A number is assigned to each character (symbol) that represents the character in the computer. A string is defined to be one or more characters. A string may be a message, a name, an address, etc. The PET uses two character sets. One set is defined for the POKE statement while the other character set is defined for the PRINT statement.

You may switch back and forth between graphics and lower case characters by changing memory location 59468 to a 14 for lower case or a 12 for graphics. When location 59468 contains a 14 both the PRINT and POKE statements will display lower case characters.

To switch

T: POKE 59468,14
R: READY.

Using the shift key will now give lower case characters.

T: A (when holding shift key down)
R: a
R: READY.
T: POKE 59468,12
R: READY.

Using the shift key will now give graphic characters.

To permit intercomputer communication and establish a standard, the American Standard Code for Information Interchange (ASCII) character set was defined. It includes all letters (both upper and lower case), digits, and many special symbols. Some of the characters used in BASIC are taken from the ASCII character set. The character set used with the POKE command will be described in the workbook on graphics. Both character sets are in the APPENDIX of this workbook. When we talk about strings in this workbook, we will mean the ASCII character set and its numerical representation. For example, the ASCII character A is represented by the number 65, the character B by the number 66, and the character 1 by the number 49. The numerical representation of the PET keyboard uses the numbers 0-255. The PRINT statement displays some characters and treats others as cursor control so that not all 256 characters are visible on the screen.

The string functions make the PET a powerful and versatile computer. By using the string functions the PET can truly become a personal computer.

STRINGS

It is reasonably easy to tailor your programs so that they actually carry on a dialogue with you. ANIMAL¹ and ELIZA² are examples of programs that make extensive use of strings to carry on a dialogue. Strings are also useful in computer-aided instruction and in searching library indexes.

There are several programs at the end of this workbook showing the various techniques for using strings. There are also several problems for you to solve. The best way to learn to use strings is to write programs that use strings. As you go through the examples in this section, be sure you understand what is being done before you continue. Some of the examples are easy to read and understand; other examples are more complicated. All examples should be tried on your PET.

A. Legal String Names

A string name is a variable name followed by a dollar sign (\$). A string name may be up to 6 characters, including the dollar sign (\$). Only the first two characters of the name are actually used, however. The first character must be alphabetic. If more than 6 alphabetic characters are used, a syntax error results. However, you may use at least 50 characters if they are all numeric except the first which must be alphabetic. Since only the first two characters are used in the variable name, you must be careful in your naming conventions. A string is defined by enclosing it in quotes. For example, to show that only the first two characters are used.

Exercise: Check string variable names.

```
T: NEW
T: ABCDE$="J"
T: AB123$="K"
T: PRINT ABCDE$,AB123$
R: K          K
T: PRINT AB$
R: K
T: A123456789$="S"
T: PRINT A1$
R: S
T: ABCDEFG$="B"
R: ?SYNTAX ERROR
```

7 is dec 1.

The ?SYNTAX ERROR shows that 7-character are not legal for a string variable name.

1 101 Games, David Ahl

2 Creative Computing July-Aug 1977 Vol 3, No 4

B. Subscripted String Variables

A subscripted variable need not be dimensioned if the subscript will not exceed 10. A non-subscripted variable is unique and distinct from a subscripted variable of the same name. For example, A\$ is distinct from A\$(0) or A\$(1).

Exercise: Show the independence of variable names.

```
T: NEW
T: 10 A$="A"
T: 20 A$(0)="B"
T: 30 A$(1)="C"
T: 40 A$(2)="D"
T: 50 PRINT A$;A$(0);A$(1);A$(2)
T: RUN
R: ABCD
```

Note that 0 is a valid subscript. Also note in statement 10, A\$ is not subscripted and is unique and distinct from all subscripted variables of the same name. It was not necessary to DIMension A\$(0)-A\$(3) because the subscript does not exceed 10.

Exercise: Show that string arrays greater than 10 must be dimensioned.

```
T: 10 FOR I=1 TO 11
T: 20 A$(I)="*"
T: 30 NEXT I
T: 40 END
T: RUN
R: ?BAD SUBSCRIPT ERROR IN 20
```

In the above example, A\$ must have a DIM A\$(11) statement since the subscript value does exceed 10.

C. Maximum String Length

Exercise: Determine the maximum string length.

```
T: NEW
T: FOR I= 1 TO 255:A$=A$+"D":NEXT I
R: READY
```

Note that 255 characters in a string is ok.

```
T: NEW
T: FOR I= 1 TO 256:A$=A$+"D":NEXT I
R: ?STRING TOO LONG ERROR
```

Note that 256 characters in a string is invalid.

IV. OPERATIONS ON STRINGS

A string variable may be lengthened by concatenation. Concatenate means to attach.

Exercise: Concatenate three strings.

```
T: NEW
T: 10 A$="P"
T: 20 B$="E"
T: 30 C$=A$+B$+"T"
T: 40 PRINT C$
T: RUN
R: PET
```

Let's look at the way the + sign is interpreted when used with numeric values and when used with string values.

Exercise: Show how the + sign is used with numbers and with characters.

```
T: NEW
T: 10 A$="123"
T: 20 B$="456"
T: 30 A=123
T: 40 B=456
T: 50 C$=A$+B$
T: 60 C=A+B
T: 70 PRINT C$,C
T: RUN
R: 123456 579
```

*→ die 1 immer getallen. " ? die 4-3
hierina
in prozessoren meist distinkt*

*+ < > = am f.
möglicherweise
if A\$ < B\$ THEN...
die Variable 5-9*

In statement 50, the string B\$ is concatenated (attached) to the right-hand side of the string in A\$. In statement 60, a numeric addition is made.

Strings need to be manipulated in a different manner than numeric values. The + sign is the only arithmetic operator that is valid to use with character strings. However, the relational operators >, <, and = sign do work with strings in the following manner. A character is less than, equal to, or greater than the numerical representation of the character it is being compared to. Since the numerical representation of B is 66 and A is 65, A is less than B. The Boolean (logical) operators AND, OR, and NOT are invalid when used with strings.

Exercise: Show that the logical operators are not allowed with strings.

```
T: C$=A$ OR B$
R: TYPE MISMATCH ERROR
```

A. Statements

1. INPUT is used to input strings or numeric values from the keyboard. INPUT works essentially the same when inputting string characters as it does when inputting numeric values. For example:

✓ doch D ABC < P @ want A < P

PET STRING AND ARRAY HANDLING

```
T: NEW
T: 10 INPUT A$,A
T: 20 PRINT AS,A
T: RUN
R: ?
```

At this point you may input up to 72 characters, a comma, and a numeric value. The comma separates the two input values.

```
T: ATEST=,10
R: ATEST= 10
R: READY.
```

In the previous example you can put in the string you want, then hit 'RETURN'. The system will then prompt with another ? and wait until you input the numeric value. Each value may be entered on a separate line by hitting a 'RETURN' for each entry or on the same line by using commas between each value.

Exercise: Use the above program to input values a line at a time.

```
T: RUN
R: ?
T: ATEST=
R: ?
T: 10
R: ATEST= 10
```

2. DATA, READ, RESTORE. Data may be put in a program by the DATA statement and a corresponding READ statement. A DATA statement must have a corresponding value for each variable read. However, a RESTORE will restore a DATA statement so that it may be read more than once.

Exercise: Use DATA, READ, and RESTORE statements to get strings into your program.

```
T: NEW
T: 10 DATA "A","B" } verschil? zie u-3
T: 20 DATA C,D      nu.
T: 30 READ A$,B$
T: 40 PRINT A$,B$
T: 50 READ C$,D$
T: 60 PRINT C$,D$
T: 70 RESTORE
T: 80 GO TO 30
T: RUN
R: A    B
R: C    D
R: A    B
R: C    D
```

.
.
.
.

OPERATIONS ON STRINGS

The program is now in a continuous loop because RESTORE allows the DATA statements to be read over and over.

```
T: STOP
R: BREAK IN 40 (This statement number depends on when the
T: 70          STOP key is hit.)
T: RUN
R: A      B
R: C      D
R: ?OUT OF DATA ERROR IN 30
R: READY.
```

When the RESTORE is removed (statement 70) the DATA statements are only "read" once. Notice that in a DATA statement a string character does not have to be enclosed in quotes. See statement 20.

3. GET. The GET statement may be used with either numeric or string variables. Get A\$ will accept any single character typed from the keyboard.

Exercise: Input a character from the keyboard during program execution.

```
T: NEW
T: 10 FOR I=1 TO 10
T: 20 GET A$:IF A$="" GO TO 20
T: 30 PRINT A$
T: 40 NEXT I
T: RUN
T: A
R: A
```

*stop na 10 karakter
intöcken*

Statement 20 is set up for a continuous wait loop until a character is typed from the keyboard. The double quote ("") is used because it is the null or empty value of the string. As soon as a character is typed on the keyboard, it will be displayed on the screen.

```
T: 'STOP'
R: BREAK IN 20
```

B. Functions

1. ASC. The ASC function returns the numeric value corresponding to the ASCII character of the first character in a string. This function may be used to find the numeric values of the PET character set.

PET STRING AND ARRAY HANDLING

Exercise: Determine the numerical value of a PET character.

```
T: NEW
T: 10 FOR I=1 TO 1000
T: 20 GET A$:IF A$="" GO TO 20
T: 30 A=ASC(A$)
T: 40 PRINT A$,A
T: 50 NEXT I
T: RUN
T: A
R: A      65
T: any character
R:
T: 'STOP'
R: BREAK IN 20
```

The numeric value you get using this code will give a different character when used with the POKE statement. For example, if an A is typed using the above code, the integer that is returned is 65.

For example:

```
T: POKE 33728,65
R: ♦
```

This character is the upper case character of the A key. A list of both numerical representations of the PET character set may be found in the APPENDIX.

2. CHR\$. The CHR\$ function returns a character corresponding to the integer given. The integer value must be between 0 and 255. The CHR\$ function is the inverse of the ASC function, that is, it converts in the opposite direction. The CHR\$ function is given an integer value and returns the corresponding character. The ASC function is given a character and returns the corresponding integer value.

Exercise: Display the numerical equivalent of each character.

```
T: NEW
T: 10 FOR I=1 TO 255
T: 20 B$=CHR(I) C=ASC(B$)
T: 30 PRINT I,B$
T: 40 GET A$:IF A$="" GO TO 40
T: 50 NEXT I
T: RUN
R: 1
T: any key
```

OPERATIONS ON STRINGS

This program will display an integer code (0-255) and the character corresponding to that integer (only the integers 33-127 and 161-255 will display a character on the screen in this code) each time you type any key on the keyboard.

Exercise: Show that CHR\$ is the inverse of ASC.

```
T: NEW
T: 10 FOR I=0 TO 255
T: 20 B$=CHR$(I)
T: 30 A=ASC(CHR$(I))
T: 40 PRINT A,B$,I
T: 50 GET A$:IF A$="" GO TO 50
T: 60 NEXT I
T: RUN
R:          0          0
T: any key
.
.
.
```

Statement 20 converts the value of I to its equivalent PET character. Statement 30 converts the PET character back to its numeric value. Statement 50 is a loop to allow you to control the rate at which things are displayed.

Exercise: Demonstrate the type of CHR\$.

```
T: NEW
T: 10 B=CHR$(5)
T: 20 PRINT B
T: RUN
R: ?TYPE MISMATCH ERROR IN 10
```

Remember that CHR\$ returns a character and that B is a real variable. To help remember the type of variable to use, remember ASC has no (\$) dollar sign, therefore, the variable name must not have a (\$) sign. Since CHR\$ has a dollar sign (\$) the variable name must also have a dollar sign (\$).

B\$ = CHR\$(40)

C = ASC("B")

3. LEFT\$. The LEFT\$ function is used to select characters from the left-hand portion of a string. To use LEFT\$, give the string name and a count of the characters chosen from the left side of the string.

PET STRING AND ARRAY HANDLING

Exercise: Select characters from a string.

```
T: NEW
T: 10 A$="ABCDE"
T: 20 B$=LEFT$(A$,1)
T: 40 PRINT B$
T: RUN
R: A
R: READY.
T: 20 FOR I = 1 TO 5
T: 30 B$=LEFT$(A$,I)
T: 50 NEXT I
T: RUN
R: A
R: A B
R: ABC
R: ABCD
R: ABCDE
R: READY.
```

*fallen kuggjeda
extra ABCDE'S.*

Don

Caution: No error message is given if I is greater than 5, that is, greater than the number of characters in A\$.

4. RIGHT\$. The RIGHT\$ function is used to select a character from the right-hand portion of a string. It works just like LEFT\$ except it selects characters starting from the right-hand side. To demonstrate this, just change LEFT\$ to RIGHT\$ in the previous example.

Don

Caution: No error message is given if you exceed the length of the string.

5. MID\$. The MID\$ function is used to select any character or group of characters from a string. MID\$ is the most powerful function of the three functions LEFT\$, RIGHT\$ and MID\$. To use MID\$, just give the string name, the number of the character you want to start with, and the number of characters you want to select.

OPERATIONS ON STRINGS

Exercise: Select characters from a string.

```
T: NEW
T: 10 A$="ABCDE"
T: 20 B$=MID$(A$,2,2)
T: 30 PRINT B$
T: RUN
R: BC
R: READY.
T: 20 FOR I = 1 TO 5
T: 30 B$=MID$(A$,1,I)
T: 40 PRINT B$
T: 50 NEXT I
T: RUN
R: A
R: AB
R: ABC
R: ABCD
R: ABCDE
R: READY.
```

```
T: NEW
T: 5 J=0
T: 10 A$="ABCDE"
T: 20 FOR I = 5 TO 1 STEP -1
T: 30 J=J+1
T: 40 B$=MID$(A$,I,J)
T: 45 PRINT B$
T: 50 NEXT I
T: RUN
R: E
R: DE
R: CDE
R: BCDE
R: ABCDE
R: READY.
```

Caution: No error message is given if I or J or both are greater than the number of characters in A\$.

DOES!

PET STRING AND ARRAY HANDLING

Exercise: Demonstrate the use of LEFT\$, RIGHT\$, and MID\$ to convert a sentence to pig Latin.

```

T: NEW
T: 10 PRINT"GIVE ME A SENTENCE"
T: 15 DIM B$(19)
T: 20 INPUT A$
T: 30 J=1 :REM J POINTS TO THE NEXT CHARACTER
T: 50 FOR I=J TO 72
T: 60 FOR K=1 TO 19
T: 80 B$(K)=MID$(A$,J+K-1,1) : REM GET NEXT CHARACTER
T: 85 IF B$(K)="" THEN I=72 : REM SET UP TO STOP
T: 90 IF B$(K)="b" GO TO 120 : REM CHECK FOR WORD BOUNDARY
T: 95 IF I=72 GO TO 120
T: 100 C$=C$+B$(K)
T: 110 NEXT K
T: 120 J=J+K
T: 130 D$=LEFT$(C$,1)
T: 150 E$=MID$(C$,2,K-2)
T: 160 F$=F$+E$+D$+"Ab"
T: 170 C$=""
T: 180 NEXT I
T: 190 PRINT F$
T: RUN
R: GIVE ME A SENTENCE
R: ?
T: CONVERT THIS TO PIG LATIN
R: ONVERTCA HISTA OTA IGPA ATINLA
R: READY.

```

Analyseren

100 C\$=C\$+B\$(K) ← verhoogd teller j, dan via 180 naar 50 want na
 het 20e karakter wordt niet gelezen
 150 E\$=MID\$(C\$,2,K-2) → gaat fout bij input 19 kar. dan is k-2 = -1
 160 F\$=F\$+E\$+D\$+"Ab" ← ween k=1 wordt
 170 C\$="" ← indien opnieuw
 180 NEXT I → binnen van meer dan 72 worden toch gepakt om instructie 95 als door 180 i=72 is. (Werkwoord aantal = 80)

6. STR\$. The STR\$ function is used to convert a number to its string equivalent. If the number has more than 9 digits, rounding takes place and the value is converted to exponential form.

MID\$?

STR\$ could be used to line up the decimal points in a check balancing program.

OPERATIONS ON STRINGS

Exercise: A program to line up the decimal point.

```
T: NEW
T: 10 DATA"10.03","100.25",".25","6.13","0"
T: 20 READ A$
T: 30 IF A$="0" GO TO 120
T: 40 FOR I=1 TO LEN(A$)
T: 50 IF MID$(A$,I,1)="." GO TO 70
T: 60 NEXT I
T: 70 FOR J=1 TO 10-I
T: 80 A$="b"+A$
T: 90 NEXT J
T: 100 PRINT A$
T: 110 GO TO 20
T: 120 END
T: RUN
R: 10.03
R: 100.25
R: .25
R: 6.13
```

*U.a. wege g getallen te display.
er kunnen nu hoogslen & cijfers
voor de komma voorkomen
Door 10 - i te vervangen
door x - i kunnen er x + 2
getallen voor de komma.*

Exercise: Convert a number to its string equivalent.

Doen

```
T: A=1234567885123
T: PRINT STR$(A)
R: 1.23456789E+12
```

Van verschil met print A.

7. VAL. The VAL function is used to convert a string containing numeric characters to a numeric value. This function is the inverse of the STR\$ function. If there are characters, other than digits in the string variable, only those digits to the left of the first non-numeric character will be converted to a numeric value. If the number is greater than 1E+38, an overflow error results.)

Exercise: Convert a string to its numerical equivalent.

```
T: NEW
T: 10 A$="1234"
T: 20 B$="A1234"
T: 30 C$="1234B"
T: 40 A=VAL(A$)
T: 50 B=VAL(B$)
T: 60 C=VAL(C$)
T: 70 PRINT A,B,C
T: RUN
R: 1234 0 1234
```

*Neem ook 123 A of B 67.
123 wordt afgedrukt.*

Statement 10 is converted as you would expect. Statement 20 is set to zero since the first character is a non-digit. Statement 30 is converted up to the first non-digit.

PET STRING AND ARRAY HANDLING

8. LEN. The LEN function counts the number of characters in a string and returns that number.

Exercise: Use the LEN function to split a string into two strings of equal or almost equal length.

```
T: NEW
T: 10 INPUT A$
T: 20 A=LEN(A$)
T: 30 B=INT(A/2)
T: 40 IF A=1 GO TO 50
T: 40 B$=LEFT$(A$,B)
T: 50 C$=RIGHT$(A$,A-B)
T: 60 PRINT B$,C$
T: RUN
R: ?
```

Input a string of your choice.

Exercise: A program to input any word and tell if it is spelled the same backward as forward.

```
T: NEW
T: 10 INPUT A$
T: 15 C$= ""
T: 20 FOR J=LEN(A$) TO 1 STEP-1
T: 30 C$=C$+MID$(A$,J,1)
T: 40 IF C$=A$ THEN 80
T: 50 NEXT J
T: 60 PRINT A$;" IS NOT THE SAME SPELLED BACKWARD "
T: 70 GO TO 10
T: 80 PRINT A$;" IS THE SAME SPELLED BACKWARD "
T: 90 GO TO 10
T: RUN
R: ?
T: KOOK
R: KOOK IS THE SAME SPELLED BACKWARD
R: ?
```

V. EXERCISES

The following problems are to check yourself to see how well you understand strings and arrays.

1. Write a program that will input a string and print out the number of characters in the string.
2. Write a program that will input a day of the week and tell whether or not it is a weekend day.
3. Write a program that will input a string and print it out in reverse order
4. Write a program that will count the number of words in a string. A word is defined to be any sequence of one or more alphabetic characters delimited by 'SPACE', coma, period, exclamation mark, colon, or end of line.
5. Write a program that will input the time of day in the form 11;15 A.M. or 8;20 P.M. and print out the time of day based on the 24 hour clock. 11;45 A.M. = 11~~45~~ HOURS: 8;20 P.M. = 20~~20~~ HOURS.
6. Write a program to input a string and print out how many vowels ^{klinders} it contains and in what positions.
7. Write a program to input a string and change all J's to H's.
8. Write a program that changes every letter in the alphabet to another letter. For example, A's might become C's and C's might become Z's, etc.
9. Write a program that will input any letter in the alphabet and print out the number 1-26 to which it corresponds.
10. Write a program to read a sentence and squeeze out all blanks.
11. Write a program to sort character strings in alphabetic order.

Examples of programs to solve these problems can be found at the end of this section. Try to write these 11 programs yourself. When you have finished compare your programs to the ones shown. Test data is provided for you to check how well your program works. Try to allow for all legal and illegal input.

PET STRING AND ARRAY HANDLING

```

1.      5  REM INPUT A STRING AND PRINT OUT THE NUMBER OF CHARACTERS
        7  REM IN IT
       10  PRINT "TYPE A STRING"
       20  INPUT A$
       30  L=LEN(A$)
       40  PRINT L
       50  GO TO 10
      100  REM TEST DATA
      110  REM INPUT A SINGLE CHARACTER (TEST END POINT)
      120  REM INPUT A LONG LINE WITH EXACTLY 38 CHARACTERS IN IT (TEST
      125  REM END POINT OF INPUT)
      130  REM ""(TEST NULL STRING)
      140  REM TEST A LINE EXACTLY 39 CHARACTERS LONG.  CLEAR THE SCREEN
      143  REM AND USE THE DOWN CURSOR SO
      145  REM THAT THE INPUT REQUEST IS ON THE BOTTOM LINE, I.E. TEST
      146  REM THE EFFECT OF SCROLLING.
      150  REM INPUT JUST A 'RETURN'
      160  REM TEST A LINE EXACTLY 78 CHARACTERS LONG.
      170  REM TEST A LINE 80 CHARACTERS LONG.  AFTER THIS TEST, 'STOP',
      175  REM 'RETURN' THEN PRINT A$
      180  REM TRY A STRING WITH A COMMA (,), THEN TRY A COLON (:)
      190  REM THE LAST 5 ARE LIMITATIONS OF THE PET INPUT STATEMENT.
      195  REM YOU CANNOT AVOID THEM UNLESS YOU USE GET$.
      210  END

```

```

2.      5  REM A PROGRAM TO INPUT A DAY OF THE WEEK AND TELL IF IT IS A
        7  REM WEEK DAY
       10  INPUT A$
       20  IF A$="SUNDAY" OR A$="SATURDAY" GO TO 80
       30  IF A$="MONDAY" OR A$="TUESDAY" GO TO 100
       40  IF A$="WEDNESDAY" OR A$="THURSDAY" GO TO 100
       50  IF A$="FRIDAY" GO TO 100
       60  PRINT A$,"IS NOT A DAY OF THE WEEK"
       70  GO TO 10

```

Statements 20 through 50 check the input against the days of the week branching to the appropriate message when a comparison is found. Statement 60 is an error message because no comparison was found. Statement 70 gives another chance for a correct input.

```

      80  PRINT A$,"IS A WEEKEND DAY"
      90  GO TO 10
     100  PRINT A$,"IS A WEEK DAY"
     110  GO TO 10
     200  REM TEST DATA
     210  REM TRY ALL 7 LEGAL DAYS
     220  REM TRY AN ILLEGAL DAY
     230  REM TRY""
     240  REM TRY 'RETURN'
     250  END

```

EXERCISES

3. 5 REM A PROGRAM TO INPUT A STRING AND PRINT IT OUT IN
 7 REM REVERSE ORDER. *moet.*
 10 INPUT "INPUT A STRING"; A\$
 20 B\$="" *of nul letter van B\$*
 25 IF LEN(A\$)=0 GO TO 60

Check for null input.

30 FOR I=LEN(A\$) TO 1 STEP -1

Start at the end of the string and decrease I by one each time through the loop.

40 B\$=B\$+MID\$(A\$,I,1)

Each time through the loop add a character to B\$. We start with I set to the position of the right most character. We then decrease the position count by 1 each time. The MID\$ function will pick one character each time. In other words, put characters from the end of the input string on the beginning of the output (reversed) string.

50 NEXT I
 60 PRINT "REVERSED STRING IS: "; *zie 5-6*
 70 PRINT B\$ *voegt even dat achter de dubbele punt*
 80 GO TO 10 *direct B\$ afgedrukt wordt.*
 100 REM TEST DATA
 110 REM INPUT A SINGLE CHARACTER (TEST END POINT) *Anders hoort B\$ op nieuwe regel.*
 120 REM INPUT A LONG LINE WITH EXACTLY 23 CHARACTERS IN IT (TEST
 125 REM END POINT OF INPUT)
 130 REM "" (TEST NULL STRING)
 140 REM TEST A LINE EXACTLY 24 CHARACTERS LONG. CLEAR THE SCREEN
 143 REM AND USE THE DOWN CURSOR SO
 145 REM THAT THE INPUT REQUEST IS ON THE BOTTOM LINE, I.E. TEST
 146 REM THE EFFECT OF SCROLLING.
 150 REM INPUT JUST A 'RETURN'
 160 REM TEST A LINE EXACTLY 63 CHARACTERS LONG.
 170 REM TEST A LINE 64 CHARACTERS LONG. AFTER THIS TEST, 'STOP',
 175 REM 'RETURN' THEN PRINT A\$
 177 REM TRY A STRING WITH A COMMA (,), THEN TRY A COLON (:)
 180 REM THE LAST 5 ARE LIMITATIONS OF PET INPUT STATEMENT. YOU
 185 REM CANNOT AVOID THEM UNLESS YOU USE GET\$.
 200 END

4. 5 REM COUNT THE WORDS IN A STRING
 8 B\$=" ,.:!"
 10 PRINT "TYPE A STRING"
 20 INPUT A\$
 24 N=0
 26 S=0

PET STRING AND ARRAY HANDLING

```

30 L=LEN(A$)
31 IF L=0 THEN 50
32 FOR I=1 TO L
33   FOR J=1 TO LEN(B$)
34     IF MID$(A$,I,1)<>MID$(B$,J,1) THEN 44
35     IF I-S>1 THEN N=N+1
36     S=I
37   GO TO 46
38 NEXT J
39 NEXT I
40 IF S<>L THEN N=N+1
41 PRINT N;" IS NUMBER OF WORDS"
42 GO TO 10
100 REM TEST DATA
110 REM INPUT A SINGLE CHARACTER (TEST END POINT)
120 REM INPUT A LINE OF ALL SEPARATORS .,! ETC.
130 REM "" (TEST NULL STRING)
140 REM TEST WORD WITH LOTS OF TRAILING SEPARATORS
150 REM INPUT JUST A 'RETURN'
160 REM TEST A WORD EXACTLY 78 CHARACTERS LONG.
170 REM TEST A WORD 80 CHARACTERS LONG. AFTER THIS TEST, 'STOP',
175 REM 'RETURN' THEN PRINT A$

```

is $P-S=1$ dan is $P=1$ en $S=2$
 dus eerste teken is
 spatie . , of ! of !
 Algemeen $i-S$ is nodig
 om onderscheid te
 kunnen maken tussen be-
 : AB. en :.
 hier is $i-S=2$

$S \neq L \Rightarrow$ de laatste
 tekens vormen
 een woord.

bu. 5. 5 REM CONVERT TIME TO THE 24 HOUR CLOCK
 voer in 10 INPUT "WHAT IS THE TIME" ; A\$
 AB;CD;EF A.m. 20 L=LEN(A\$)
 30 IF L > 6 THEN 60 test op te weinig karakters = gewond
 40 PRINT "BAD TIME"
 50 GO TO 10
 60 B\$=RIGHT\$(A\$,4)
 70 IF(B\$<>"A.M.") AND (B\$<>"P.M.") THEN 40 trivial
 80 H=VAL(A\$)
 90 IF H<0 OR H>12 THEN 40 } trivial.
 100 FOR I=1 TO L-4 } wenn = H.
 110 IF MID\$(A\$,I,1)=";" THEN 140

The semicolon is chosen as a separator instead of the colon because the INPUT statement thinks the colon is a delimiter.

m = CD.
 S = EF.
 120 NEXT I test op ; vergeten
 130 GO TO 40
 140 C\$=RIGHT\$(A\$,L-I)
 150 M=VAL(C\$) } minuten = m
 160 IF M<0 OR M>59 THEN 40
 170 FOR J=I+1 TO L-4
 180 IF MID\$(A\$,J,1)=";" THEN 210
 190 NEXT J
 200 GO TO 40
 210 D\$=RIGHT\$(A\$,L-J)
 220 S=VAL(D\$) } seconden = S
 230 IF S<0 OR S>59 THEN 40
 250 IF B\$="P.M." THEN H=H+12

EXERCISES

260 TP\$=STR\$(H) *in tikkie bu*
 270 IF LEN(TP\$) < 3 THEN TP\$=" 0" +RIGHT\$(TP\$,1) *H = 25. TP\$ = STR\$(H) een wege.*
 280 T\$=TP\$ *why?*
 290 TP\$=STR\$(M) *N.B. een getal van n cijfers een een string-lengte n+1 want 1e cijferi display staat op*
 300 IF LEN(TP\$) < 3 THEN TP\$=" 0" +RIGHT\$(TP\$,1)
 310 T\$=T\$+" "+RIGHT\$(TP\$,2)
 320 TP\$=STR\$(S)
 330 IF LEN(TP\$) < 3 THEN TP\$=" 0" +RIGHT\$(TP\$,1)
 340 T\$=T\$+" "+RIGHT\$(TP\$,2)
 350 PRINT " 24 HOUR TIME IS ";T\$ *werkt hetzelfde als in ub. 3 blz 5-3*
 360 GO TO 10
 400 REM TEST DATA
 410 REM TRY THE FOLLOWING LEGAL DATA
 420 REM TRY 07;15;00 A.M.
 430 REM TRY 07;15;A.M.
 440 REM TRY 07;;A.M.
 450 REM TRY 07;15;00 P.M.
 460 REM TRY 07;15;P.M.
 470 REM TRY 07;;P.M.
 480 REM TRY 7;15;0 A.M.
 490 REM TRY 7;15;P.M.
 500 REM TRY 0;01;00 A.M.
 510 REM TRY ;;;P.M.
 520 REM TRY 0;0;05 A.M.
 530 REM TRY 0;;3P.M.
 540 REM TRY THE FOLLOWING ILLEGAL DATA
 550 REM TRY 13;05;01 A.M.
 560 REM TRY 1;15;01 X.Y.
 570 REM TRY 5;61;11 A.M.
 580 REM TRY 3;-4;2 P.M.
 590 REM TRY 3;4;-1 A.M.
 600 REM TRY 2;4;60 P.M.
 610 REM TRY -1;5;1 A.M.

6.

1 REM A PROGRAM TO COUNT THE VOWELS IN A STRING
 5 DIM PL(78)
 10 INPUT "INPUT A STRING";A\$
 15 N=0
 17 FOR I=1 TO 78:PL(I)=0:NEXT I
 20 B\$="AEIOU"
 30 FOR I=1 TO LEN(A\$)
 40 FOR J=1 TO 5
 50 IF MID\$(A\$,I,1) <> MID\$(B\$,J,1) GO TO 80

klinkers

string niet langer dan 77 karakteren

The outer loop is used to check each character in A\$ while the inner loop is to check each character in B\$. Since non-vowels are not interesting, simply branch to the end of the inner loop.

PET STRING AND ARRAY HANDLING

```
60 N=N+1
70 PL(I)=1
```

N is used as a counter for the total number of vowels.
The array PL is used to calculate the position of each vowel found. Each position that has a vowel is set to 1.

```
80 NEXT J
90 NEXT I
100 PRINT "NUMBER OF VOWELS ARE";N;"POSITIONS ARE";
```

Since the total number of vowels is to be printed only once, it is done in statement 100. At the same time set up to print the positions on the same line by ending the PRINT statement with a;

plaats de gevonden posities direct achter elkaar op volgende regel

```
110 FOR J=1 TO 78
120 IF PL(J)=0 GO TO 140
130 PRINT J;
140 NEXT J
```

The above loop is set up to find the positions of each vowel; that is, if PL(J)=1 then J is the position number of a vowel. The PRINT statement at 130 will print on the same line as statement 100 each time through the loop because of the semicolon (;) at the end of the statement.

vgl. blz 5-3

```
145 PRINT
146 PRINT
150 GO TO 10
```

*The two lines
→ schrijft twee regels op*

```
200 REM TEST DATA
210 REM INPUT A SINGLE CHARACTER
220 INPUT A LONG LINE WITH ALL VOWELS IN IT
230 REM INPUT "" (TEST A NULL STRING)
240 INPUT JUST A 'RETURN'
250 REM TRY A STRING WITH NO VOWELS IN IT
260 END
```

```
7. 5 REM A PROGRAM TO CHANGE ALL J's TO H's.
10 INPUT "INPUT A STRING";A$
20 L=LEN(A$)
30 FOR I=1 TO L
40 B$=MID$(A$,I,1)
50 IF B$="J" THEN B$="H"
60 C$=C$+B$
70 NEXT I
80 PRINT "NEW STRING IS ";C$
100 REM TEST DATA
110 REM INPUT A SINGLE J
120 REM INPUT A SINGLE CHARACTER
130 REM INPUT A "" (TEST A NULL STRING)
140 REM INPUT A STRING WITH A J AS THE LAST CHARACTER
150 END
```

EXERCISES

8. 10 REM A PROGRAM TO CHANGE EVERY LETTER IN THE ALPHABET TO
 15 REM ANOTHER LETTER.
 20 INPUT "TYPE A STRING"; A\$
 25 C\$=""
 30 FOR I=1 TO LEN(A\$)
 40 B\$=MID\$(A\$,I,1)
 45 IF B\$= "" GO TO 90
 50 B=ASC(B\$)-3

Statement 40 gets a character. Statement 50 converts the character to its equivalent number and then subtracts 3 in order to change it to a different character.

60 IF B >= 65 GO TO 80 *A t/n 2 is 65 t/n go.*
 70 B=B+26

Since the alphabetic characters are the numbers between 65 and 90, adjustments must be made if the character is less than 65. By adding 26 only when it is less than 65, we can never exceed 91. However, the case where the input characters do not fall in the range 65 - 90 is not provided for.

80 C\$=C\$+CHR\$(B)

Add the new character to the string.

90 NEXT I
 100 PRINT C\$
 110 REM TEST DATA
 120 REM INPUT A SINGLE CHARACTER (TEST END POINT)
 130 REM INPUT A LONG LINE WITH EXACTLY 24 CHARACTERS IN IT
 135 REM (TEST END POINT OF INPUT)
 140 REM ""(TEST NULL STRING)
 150 REM TEST A LINE EXACTLY 25 CHARACTERS LONG.
 155 REM TEST A LINE OF SPECIAL CHARACTERS
 160 REM INPUT JUST A 'RETURN'
 170 REM TEST A LINE EXACTLY 64 CHARACTERS LONG.
 180 REM TEST A LINE 65 CHARACTERS LONG. AFTER THIS TEST, 'STOP',
 185 REM 'RETURN' THEN PRINT A\$
 190 REM TRY A STRING WITH A COMMA (,), THEN TRY A COLON (:)
 200 ~~REM THE LAST 5 ARE LIMITATIONS OF THE PET INPUT STATEMENT~~
 205 ~~REM YOU CANNOT AVOID THEM~~
 210 ~~REM UNLESS YOU USE GET\$.~~
 220 END

PET STRING AND ARRAY HANDLING

```

9.      5  REM A PROGRAM TO COMPUTE THE ALPHABETIC POSITION OF A CHARACTER.
        8  PRINT "ENTER CHARACTER"
       10  INPUT A$
       15  IF LEN(A$) > 1 THEN PRINT "ENTER SINGLE CHARACTER"; GO TO 10
       20  FOR I=1 TO 26
       30  B$="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
       40  IF A$=MID$(B$,I,1) GO TO 70
       50  NEXT I
       60  PRINT "NOT AN ALPHABETIC CHARACTER"
       65  GO TO 8
       70  PRINT A$;" IS IN POSITION";I
       80  GO TO 8
       90  END
      100  REM TEST DATA
      110  INPUT A SINGLE CHARACTER
      120  REM INPUT A LONG LINE
      130  REM "" (TEST NULL STRING)
      140  REM TEST A SPECIAL CHARACTER

```

```

10.     5  REM A PROGRAM TO SQUEEZE OUT BLANKS.
        10  INPUT A$
        15  B$=""
        18  IF LEN(A$)=0 THEN 60
        20  FOR I=1 TO LEN(A$)
        30  IF MID$(A$,I,1)=" " GO TO 50
        40  B$=B$+MID$(A$,I,1)
        50  NEXT I
        60  PRINT "SQUEEZED STRING:"
        65  PRINT B$
        70  GO TO 10
      100  REM TEST DATA
      110  INPUT A SINGLE CHARACTER (TEST END POINT)
      120  REM INPUT A LINE WITH NO BLANKS IN IT
      130  REM "" (TEST NULL STRING)
      140  REM TEST A LINE WITH ALL BLANKS
      150  REM INPUT JUST A 'RETURN'
      180  REM TRY A STRING WITH A COMMA (,), THEN TRY A COLON (:)
      210  END

```

EXERCISES

Voor ander programma zie notes
zie blz 38 Computerwetkunde

```

11. 5 REM EXAMPLE OF A PROGRAM TO SORT CHARACTER STRINGS IN
    6 REM ALPHABETIC ORDER USING HEAPSORT.
    10 DIM R$(100)
    100 PRINT "PLEASE INPUT THE NAMES YOU WANT SORTED"
    110 PRINT "WHEN YOU ARE FINISHED INPUT EOF."
    120 N=0
    130 N=N+1
    140 INPUT R$(N)
    150 IF R$(N)<>"EOF" GO TO 130
    160 N=N-1
    170 L=INT(N/2)+1
    180 IR=N
    190 IF L=1 GO TO 212
    200 L=L-1
    210 S$=R$(L)
    211 GO TO 220
    212 S$=R$(IR)
    213 R$(IR)=R$(L)
    214 IR=IR-1
    215 IF IR<=1 GO TO 310
    220 J=L
    230 I=J
    240 J=2*J
    250 IF J=IR GO TO 265
    260 IF J>IR GO TO 290
    261 IF R$(J)>=R$(J+1) GO TO 265
    262 J=J+1
    265 IF S$>=R$(J) GO TO 290
    270 R$(I)=R$(J)
    280 GO TO 230
    290 R$(I)=S$
    300 GO TO 190
    310 R$(1)=S$
    320 FOR I=1 TO N
    330 PRINT R$(I)
    340 NEXT I
    350 END
    400 REM TEST DATA
    410 REM INPUT A SINGLE CHARACTER
    420 REM TRY 123
    430 REM TRY ""
    440 REM TRY 'RETURN'
    450 REM TRY 2 LONG STRINGS
    460 REM TRY A STRING WITH A COMMA (,) IN THE MIDDLE
    465 REM END ALL TEST CASES WITH "EOF"

```

maximaal 100 woorden
toe te laten gaan

zie blz 4-1. Alleen 1e letters
worden vergeleken

Geeft dimensie fout vanwege
als 360 9 0 10 6

Te herstellen door
to Dim R\$(147) max
zie blz 2-3.
165 Dim R\$(N)
350 9 0 10 100

VI. APPENDIX - Character Sets

The following character sets are those used by the Commodore PET. The character sets described will be those used with the POKE statement and those used with the PRINT, CHR\$, and ASC statements. There are three sets for the POKE statement, graphics, lower case, and reverse video. There are only two sets for the PRINT, CHR\$, and ASC statements. One is for graphics and the other is for lower case.

The character set used with the PRINT statement is not as straight forward as the one used with the POKE statement. For example, to get a reverse video character with the PRINT you must first hit the double quote (") then the 'RVS' key. All characters after that will be reverse video until the terminating double quote (").

Example of displaying reverse video characters.

```
T: PRINT" [R]AB"
R: [A][B]
R: READY.
```

The characters in boxes represent reverse video characters. The [R] is the character you get when the reverse video key is hit.

An alternative method to get reverse video characters is to type CHR\$(18) rather than hit the 'RVS' key.

Example of displaying reverse video characters.

```
T: PRINT CHR$(18);"ABC"
R: [A][B][C]
R: READY.
```

Some characters have a dual numeric representation when used with the CHR\$ function. For example, CHR\$(35) = CHR\$(99) = #. The following is a list of those numbers that represent the same character.

33,97	47,111	60,124	170,234	183,247
34,98	48,112	61,125	171,235	184,248
36,100	49,113	62,126	172,236	185,249
37,101	50,114	63,127	173,237	186,250
38,102	51,115	161,225	174,238	187,251
39,103	52,116	162,226	175,239	188,252
40,104	53,117	163,227	176,240	189,253
41,105	54,118	164,228	177,241	190,254
42,106	55,119	165,229	178,242	222,255
43,107	56,120	166,230	179,243	
44,108	57,121	167,231	180,244	
45,109	58,122	168,232	181,245	
46,110	59,123	169,233	182,246	

APPENDIX

A. Character Codes for the PRINT, CHR\$, and ASC.

Lower case character set (location 59468 = 14).

161	162	163	164	165	167	166	220	168	169	223
33	34	35	36	37	39	38	92	40	41	95

147	145	157	148
19	17	29	20

209	215	197	210	212	217	213	201	207	208	222
81	87	69	82	84	89	85	73	79	80	94

183	184	185	175
55	56	57	47

193	211	196	198	199	200	202	203	204	186
65	83	68	70	71	72	74	75	76	58

141
RETURN

180	181	182	170
52	53	54	42

218	216	195	214	194	206	205	172	187	191
90	88	67	86	66	78	77	14	59	63

177	178	179	171
49	50	51	43

146	192	219	221	160	188	190			
18	64	91	93	32	60	62			

176	174	173	189
48	46	45	61

B. Character Codes for the PRINT, CHR\$, and ASC.

Graphics character set (location 59468 = 12).

161	162	163	164	165	167	166	220	168	169	223
33	34	35	36	37	39	38	92	40	41	95

147	145	157	148
19	17	29	20

209	215	197	210	212	217	213	201	207	208	222
81	87	69	82	84	89	85	73	79	80	94

183	184	185	175
55	56	57	47

193	211	196	198	199	200	202	203	204	186
65	83	68	70	71	72	74	75	76	58

141
RETURN
13

180	181	182	170
52	53	54	42

218	216	195	214	194	206	205	172	187	191
90	88	67	86	66	78	77	44	59	63

177	178	179	171
49	50	51	43

146	192	219	221	160	188	190
18	64	91	93	32	60	62

176	174	173	189
48	46	45	61

C. Character Codes for the POKE Statement.

Reverse ~~vide~~^{dec} lower case character set (location 59468=14).

225 226 227 228 229 231 230 220 232 233 223



161 162 163 164 165 167 166 156 168 169 159

209 215 197 210 212 217 213 201 207 208 222



145 151 133 146 148 153 149 137 143 144 158

247 248 249 239



183 184 185 175

193 211 196 198 199 200 202 203 204 250



129 147 132 134 135 136 138 193 140 186

244 245 246 234



180 181 182 170

RETURN

218 216 195 214 194 206 205 236 251 255



154 152 131 150 130 142 141 172 187 191

241 242 243 235



177 178 179 171

192 219 221 224 252 254



128 155 157 160 188 190

240 238 237 253



176 174 173 189

D. Character Codes for the POKE statement.

Graphics character set (location 59468 = 12).

97	98	99	100	101	103	102	92	104	105	95
33	34	35	36	37	39	38	28	40	41	31

81	87	69	82	84	89	85	73	79	80	94
17	23	5	18	20	25	21	9	15	16	30

119	120	121	111
55	56	57	47

65	83	68	70	71	72	74	75	76	122
1	19	4	6	7	8	10	11	12	58

116	117	118	106
52	53	54	42

90	88	67	86	66	78	77	108	123	127
26	24	3	22	2	14	13	44	59	63

113	114	115	107
49	50	51	43

	64	91	93	96	124	126		
18.	0	27	29	32	60	62		

112	110	109	125
48	46	45	61

RETURN

E. Character Codes for the POKE statement.

Lower case character set (location 59468 = 14).

97	98	99	100	101	103	102	92	104	105	95
33	34	35	36	37	39	38	28	40	41	31

81	87	69	82	84	89	85	73	79	80	94
17	23	5	18	20	25	21	9	15	16	30

119	120	121	111
55	56	57	47

65	83	68	70	71	72	74	75	76	122
1	19	4	6	7	8	10	11	12	58

116	117	118	106
52	53	54	42

90	88	67	86	66	78	77	108	123	127
26	24	3	22	2	14	13	44	59	63

113	114	115	107
49	50	51	43

	64	91	93	96	124	126	
	0	27	29	32	60	62	

112	110	109	125
48	46	45	61

RETURN



Notes

Array 2-1

Random generator; Dobbelsteen 2-4

String 3-1

ASCII 3-1

Poker 59468, 12/14 3-1 ~~na 4~~
33720 4-4

+ > < = { bij vergelijken en
AND OR NOT koppelen alfa num info 4-1.
invoer bij INPUT 4-2.

RESTORE

GET: A\$ = "" 4-3.
vgl. A=0.

FOR -- STEP -- α 4-7.

Het onder elkaar krijgen van decimale punt 4-9.

Het zoeken naar symmetrische woorden 4-10

IF -- THEN 5-4 5-8

Print α [5] 5-2.

Het alfabetisch rangschikken 5-9.

Notes

```

10  Dim A$(1431)
20  ? " Over de namen in die je wilt sorteren
30  ? " Als je klaar bent voer dan 'eind' in
40  k = 0
50  k = k + 1
60  input A$(k)
70  if A$(k) = "eind" goto 50
80  k = k - 1
90  for j = k to 2 step -1
100  for i = 1 to j - 1
110  if A$(i) > A$(i+1) then x = 0 : goto 160
120  goto 170
130  x = 1 (geeft aan dat gewisseld is)
140  B$ = A$(i)
150  A$(i) = A$(i+1)
160  A$(i+1) = B$
170  next i
180  next j
190  if x = 1 goto 90
200  180 for i = 1 to k
210  190 print A$(i) Nummer? dan print i; " "; A$(i) (of A$(i);)
220  200 next i
230  210 print
240  220 print
250  230 END of goto 20

```

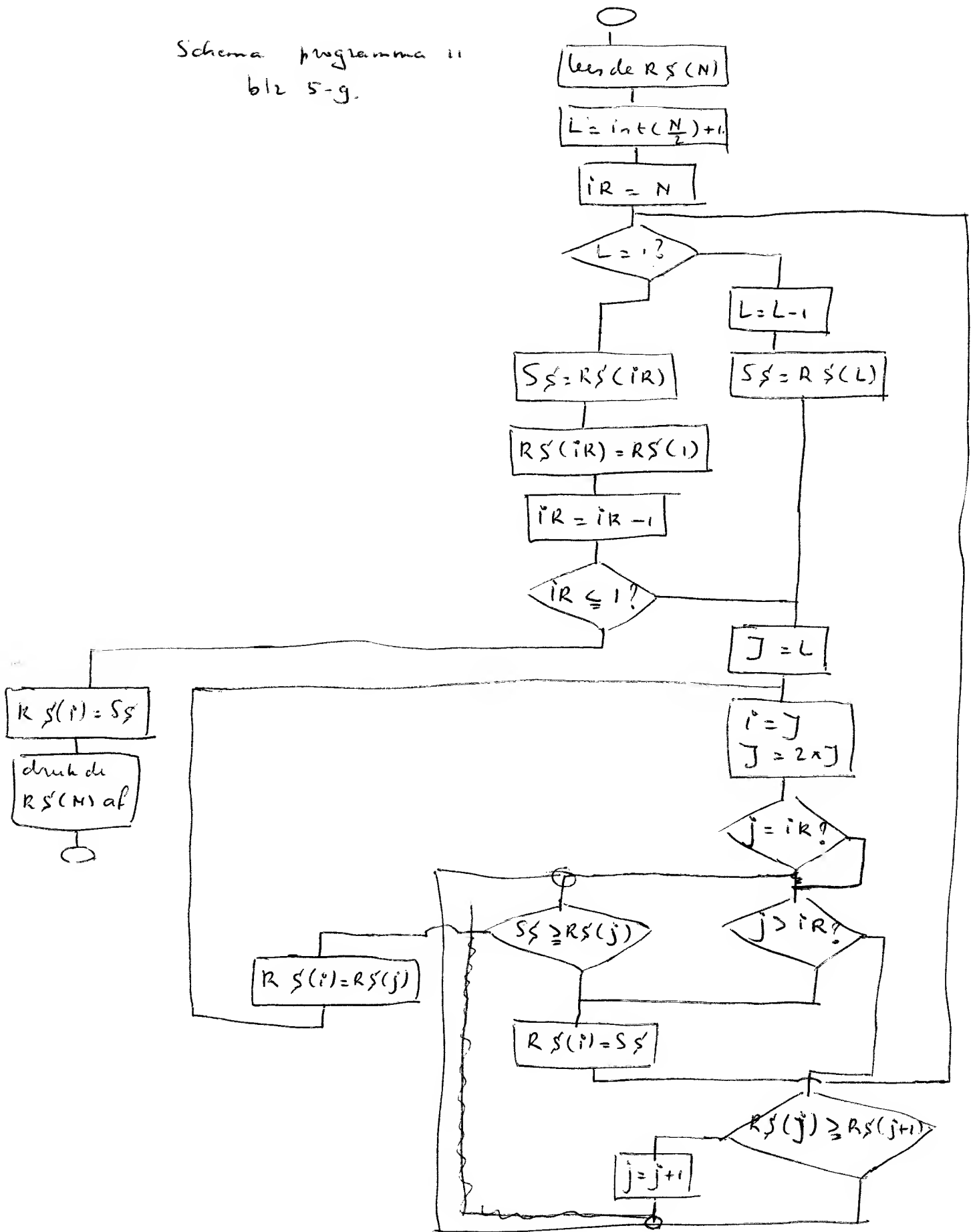
A(i) > A$(i+1)$ worden gewisseld.
 Daardoor komt de grootste op de laatste plaats (k); vandaan

↓
 voorachter
 elkaar.

Dit programma ordent ook getallen, maar dan wel getallen met evenveel cijfers

Bij rangschikken 1235 en 365 → 365, 1235
 op te vangen door 1235, 0365

Schema programma II
blz 5-g.



T even \Rightarrow N even, N odd \Rightarrow T odd

Then $1, 2, \dots, L$; $L+1, \dots, 2L-2 = N$.

Noneven $1, 2, \dots, L$; $L+1, \dots, 2L-1 = N$.

Notes